Experimento velocidad de ejecución

1. Proceso o sistema que se estudia y variables respuestas de interés

El proceso estudiado es la velocidad de ejecución de código escrito en C usando y no usando OpenMP, adicionalmente se compararán estas salidas con ejecución de código escrito en Python y R.
Definiciones básicas: OpenMP es una interfaz de programación de aplicaciones para la programación multiproceso de memoria compartida en múltiples plataformas.
Lenguajes de programación usados: C, Python (Numpy) R.
Las variables que intervienen en este experimento son:

- Computadora usada.
- Lenguaje usado.
- Algoritmo usado.

La variable respuesta corresponde al tiempo medido en segundos de una multiplicación de matrices NxN con valores entre 1 y 100 generadas aleatoriamente, estas ejecuciones se desarrollaron una vez reiniciada la computadora, desconectando el internet y cualquier evento, esperando a que todos los procesadores (CPU) estuvieran aproximadamente al 100% de sleeping.

2. Objetivo del experimento

Comparar y evidenciar existencia de reducción de tiempo usando OpenMP. Se espera que el experimento responda a las siguientes preguntas.

- ¿Existe diferencia en tiempo en la ejecución de algoritmo escrito en C con OpenMP y sin OpenMP?
- ¿Existe diferencia en tiempo en la ejecución de los algoritmos escritos en C y los algoritmos escritos en Python (Numpy) y R.

No es posible generalizar resultados a otros tipos de lenguajes a los usados; pues cada lenguaje puede diferir en su forma de ejecución. Tambien debido a las limitaciones de equipos, no se puede esperar que las condiciones de experimentación emulen las condiciones úsales a las que están ejecutando dichos algoritmos. Las conclusiones que se obtengan sólo pueden ser discutidas en términos de las condiciones puestas en este experimento, aunque ellas podrían dar indicios de los resultados que se darían bajo condiciones más normales.

3. Identificación de todas las fuentes de variación:

3.1. Factores de estudio y sus niveles

El factor de tratamiento “tipo de algoritmo” se han elegido tres lenguajes de programación: C, C con OpenMP, Python (Numpy) y R. Note que el experimentador no controla la existencia de deamons o tareas sobre el pc; asume que los pc usados serán típicos de la población disponibles en las tiendas. Otro factor de tratamiento “tamaño de matriz” se han elegido 5 tamaños de matriz: 100x100, 200x200, 300x300, 400x400 y 500x500, con valores aleatorios entre 1 y 100. Si esta presunción no es cierta, entonces los resultados del experimento no tendrán una aplicación general. Se tomarán 30 veces el tiempo por algoritmo con el fin de que el experimento sea tan representativo como pueda ser posible de la población de ejecuciones.

3.2. Factores de bloque, de ruido y covariables

A parte de las diferencias en la forma de ejecución del algoritmo en los diferentes lenguajes, los procesadores pueden variar, adicionalmente el calor generado por el pc, y la temperatura del día no fueron necesariamente todos expuestos a la misma cantidad de calor. Sin embargo, el experimentador no sintió que las unidades experimentales serían suficientemente variables como para justificar bloqueo. Estos pudieron haber sido usados como covariables, pero el experimentador eligió en cambio medir los cambios de tiempo, es decir, el tiempo final menos el tiempo inicial.
Otras fuentes de variación incluyen generación de números aleatorios, imprimir en pantalla. Todas estas fueron consideradas menores. Ningún factor de ruido fue incorporado al experimento.

3.3. Unidades experimentales

El experimento se llevará a cabo usando diferentes pc, a los que se desconecta las redes, bluetooth y cualquier otro tipo de conexión inalámbrica, los cuales se reinicia el pc, al verificar que los CPU's estén aproximadamente al 100% sleeping se ejecutan los algoritmos, estos se ejecutan por consola. Se hará un registro de los tiempos de ejecución en segundos.

Un estudio piloto indicó que esta manera de ejecución es suficiente para cubrir cualquier variación que el pc podría dar a los tiempos de ejecución de los algoritmos por los tamaños de algoritmo.

4. Regla de asignación de las unidades experimentales a los tratamientos

Un número igual de observaciones será hecha sobre cada uno de los niveles de los factores. Por tanto, se prepararán ***n*** ejecuciones por cada algoritmo y tamaño de matriz. Este tamaño elegido fue de 30 para cumplir con el **TLC**. No se aleatorizarán por facilidad de ejecución.

5. Procedimiento experimental, mediciones y dificultades anticipadas

Los algoritmos son compartidos por GitHub para la reproducibilidad del experimento, al verificar que los CPU's estén aproximadamente al 100% sleeping se ejecutan los algoritmos y desconectar las redes, bluetooth y cualquier otro tipo de conexión inalámbrica
Después que se han ejecutado los algoritmos, en pantalla se muestra el tiempo de ejecución en segundos. Estos tiempos serán registrados. El análisis se realizará sobre las diferencias en el uso de OpenMP.

Dificultades experimentales esperadas:

- Programas, tareas o deamons en el pc. Por tanto, los datos pueden no mostrar diferencias en los tiempos.
- Tiempos afectados por la ejecución anterior, correlación entre muestras.
- Tiempos mayores a los esperados en el estudio piloto.

Dificultades encontradas durante la experimentación:

- En el estudio piloto se evidencio tareas que no fueron apagadas y afectaban el tiempo como el internet o el bluetooth de los audífonos.
- El uso de consola disminuía los tiempos, usar herramientas IDE aumentaba tiempos.
- En el estudio piloto se evidencio correlaciones entre los tiempos.

6 Experimento piloto

Se corrió un experimento piloto con dos propósitos: Primero, para identificar las dificultades listadas en la Sección 5, segundo, para obtener una estimación de la varianza para estimar el número de réplicas. La varianza del error fue estimada aproximadamente de 0.146 s^2, el valor del MSE en dicho experimento piloto. De hecho, este es una sobre estimación y hubiese sido mejor usar un intervalo de confianza unilateral para la varianza.

7. Especificación del modelo estadístico, análisis o pruebas a realizar y tamaños de muestra

Dado que se tendrá cuidado en controlar todas las fuentes extrañas de variación, se asumió que el siguiente modelo es una aproximación razonable:

$$\begin{aligned} &\boldsymbol{Y_{ij}} = \mu + \alpha_i + \varepsilon_{ij}, \quad \varepsilon_{ij} \overset{iid}{\sim} \boldsymbol{N}(0, \sigma^2), \quad \boldsymbol{i} = 1,2,3; \quad \boldsymbol{j} = 1,2,3,4 \\ &\sum_{\boldsymbol{i}=1}^{3} \alpha_i = 0 \end{aligned} \tag{1}$$

Nota: Me equivoque y es i=j=2

Con el fin de responder a la pregunta sobre las diferencias en los pesos, un ANOVA de un factor de efectos fijos será calculada al 0.05 de significancia para probar:

$$\boldsymbol{H_0}: \alpha_1 = \alpha_2 = \alpha_3 = 0$$
$$\boldsymbol{H_1}: \textbf{algún } \alpha_i \neq 0 \qquad (2)$$

Nota: Me equivoque y es 1 y 2 el 3 no

Donde el estadístico de prueba y su distribución es normal y el criterio de decisión con valor P es 0.05 Si se detecta significancia estadística, se estimarán las medias y los efectos según tipo algoritmo, así como sus intervalos de confianza del 95%. Para hallar mejor las diferencias entre pares de tratamientos, se construirán los I.C de Tukey simultáneos del 95%.

Cálculo del número de observaciones necesarias: Se halla ***n=30*** para cada algoritmo y tamaño de matriz (se consideró importante detectar una diferencia en disminución de tiempos de al menos xxx% entre los algoritmos en C con OpenMP y sin OpenMP., con una probabilidad de 0.90 y nivel de significancia de 0.05).
Revisión de las decisiones anteriores: No es difícil obtener treinta observaciones de cada uno de los tratamientos; pequeños ajustes al procedimiento experimental que fueron considerados como necesarios de realizar durante el experimento piloto ya han sido incorporados.

8. Datos recolectados y análisis estadísticos de resultados

Los datos en formato csv y parquet se encuentran en el repositorio.

Vea a continuación el análisis de estos datos.

Nota: Pasar las tablas a markdown

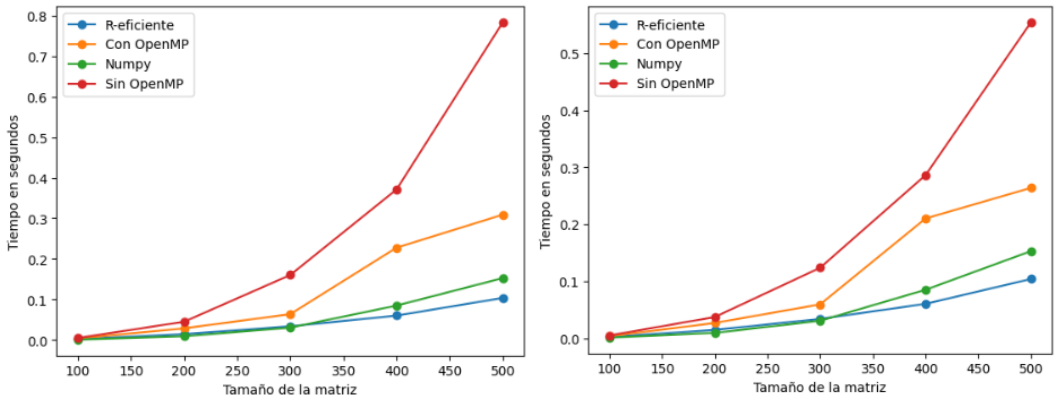

Figura 0. Distribución de la respuesta según OpenMP FxC

De la Figura 0 es claro que puede existir una diferencia visual entre las medías del tiempo medido por ejecuciones en los tipos de algoritmos FxC y FxF para los lenguajes C, Python Numpy, R

Tabla 1. ANOVA fxc

	experimentacion	estadisticos	valores_P	decision
0	100	161.445861	2.105513e-18	significativo
1	200	181.846528	1.575908e-19	significativo
2	300	5893.452207	4.963132e-60	significativo
3	400	974.006617	5.930678e-38	significativo
4	500	20827.584568	7.634794e-76	significativo

Tabla 2. ANOVA fxf

	experimentacion	estadisticos	valores_P	decision
0	100	692.124770	6.248336e-34	significativo
1	200	3413.493849	3.064264e-53	significativo
2	300	2989.318783	1.343321e-51	significativo
3	400	8518.222782	1.240523e-64	significativo
4	500	335453.914509	8.170936e-111	significativo

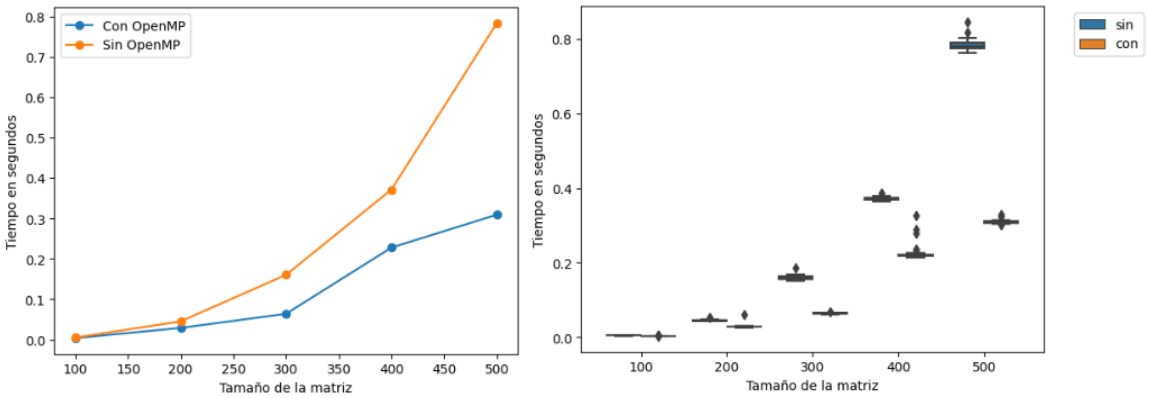

Figura 1. Distribución de la respuesta según OpenMP FxC

De la Figura 1 es claro que puede existir una diferencia significativa entre las medías del tiempo medido por ejecuciones sin OpenMP y OpenMP, lo cual es corroborado por los resultados de la prueba ANOVA en la Tabla 1 para el algoritmo que es Filas por columnas ya que Con OpecMP (línea azul) permanece por debajo en todos los experimentos, y la dispersión es mucho menor que Sin OpenMP (línea naranja). En la prueba de ANOVA, con una significancia del 5% usando el algoritmo filas por columnas, se rechazan todas las pruebas de hipótesis por lo que podemos inferir que existe una diferencia entre usar OpenMP y no usarlo.

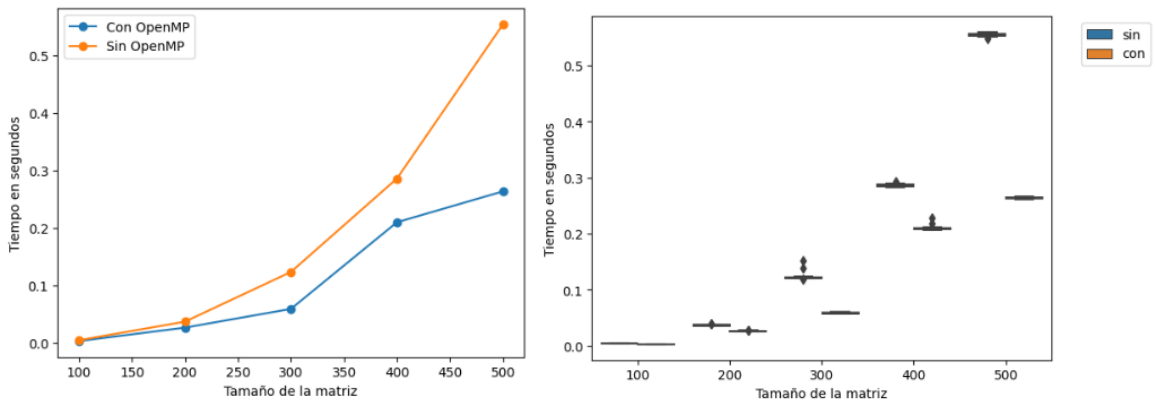

Figura 2. Distribución de la respuesta según OpenMP FxF

De la Figura 2, como ocurre en la Figura 1, con una significancia del 5% para el algoritmo usando filas por filas, existe evidencia muestral significativa se rechazan todas las pruebas de hipótesis por lo que podemos inferir que existe una diferencia entre usar OpenMP y no usarlo. Adicionalmente se observa una disminución en tiempos de xxxx.

Tabla 2. Regression

Parámetro	Valor FxC	Valor FxF
Intercepto	-0.29190067	-0.11627441
Pendiente	0.00188289	0.00080968
R2	87.1%	90.4%
MSE	0.0104	0.0014

De la tabla 2 se observa que los coeficientes para el algoritmo FxF son más pequeños eso indica que el consumo de tiempo al hacer mas grande la matriz tomara menos tiempo que usar el algoritmo FxC. La regresión para el algormitmo FxF tiene un ajuste 3.3% mayor que el algoritmo FxC, que aunque podría ser despreciable el error es mucho menor para el algoritmo FxF lo que indica que tiene menos dispersión y mas control el la toma de muestras.

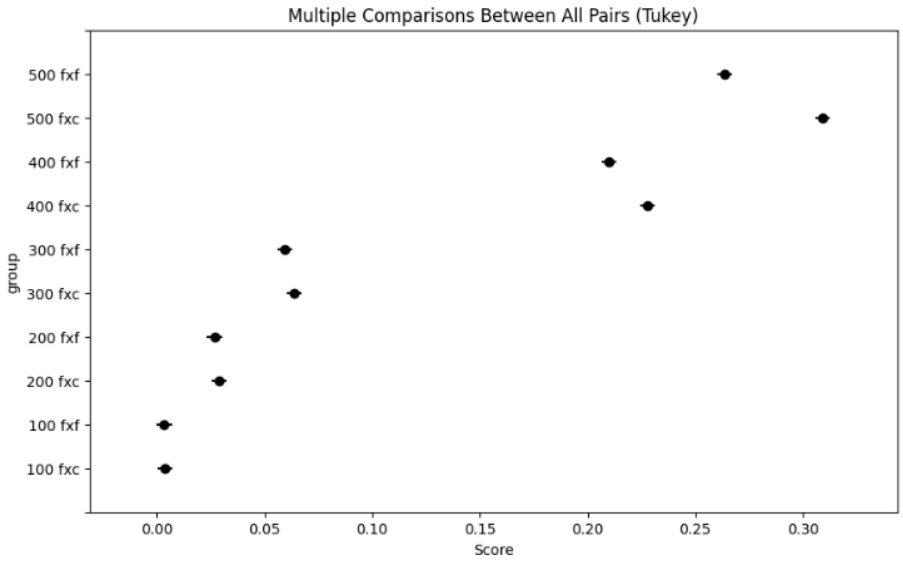

Figura 3. Distribución de la respuesta según OpenMP FxF y FxC

De la Figura 1 es el resultado grafico de la prueba de Tuckey, (el resultado numérico se encuentra en el código) con una significancia del 5% usando el algoritmo filas por columnas contra el algoritmo filas por filas, se rechaza las pruebas para las matrices igual o mayores a 300x300, lo que nos indica que existe una reducción significativa en los tiempos para la ejecución del algoritmo con OpenMP filas x filas, sin embargo en el análisis anterior, se encuentra una diferencia pequeña para las matrices iguales o mayores a 200x200 que para el algoritmo puede ser despreciable.

9. Conclusiones y recomendaciones

Con base en los resultados experimentales, los principales hallazgos fueron los siguientes:

Usando Python usando algoritmo FxC o FxF es el lenguaje más lento, lo sigue R usando el usando algoritmo FxC y FxF. Numpy es más eficiente que usar los anteriormente nombrados por lo que usa OpenMP. OpenMP para matrices pequeñas no tiene una gran ventaja, sin embargo, para matrices mayores a 200x200 es más eficiente que no usar OpenMP.La multiplicación básica de matrices en R es más efciente que los demás algoritmos y adicionalmente no usa OpenMP, solamente usa un CPU.

Como recomendación para mejorar este experimento puede considerarse lo siguiente:

Si el tiempo de ejecución no es importante para el investigador o la matriz es pequeña se recomienda usar R o C sin usar OpenMP ya que utiliza mucho tiempo escribir el código en dicho formato y en R solo es necesario 1 instrucción para dicha ejecución. Si el tiempo de ejecución es importante para el investigador o las matrices son próximas o mayores a 300x300 se recomendaría revisar la posibilidad de escribir dicho código usando C con OpenMP o utilizar numpy o R.